Cont C1

designed to store an indication of which of two different computer architectures and/or execution conventions under which instruction data of the associated page are to be executed by the processor pipeline, the indicator elements being architecturally addressable when the processor pipeline is executing under one of the architectures or data storage conventions, and architecturally unaddressable when the computer is executing under the other architecture or data storage convention;

the memory unit and/or processor pipeline further designed to recognize an execution flow from the first page, whose associated indicator element indicates the first architecture or execution convention, to the second page, whose associated indicator element indicates the first architecture or execution convention, and in response to the recognizing, to adapt a processing mode of the processor pipeline or a storage content of the memory to effect execution of instructions in the architecture and/or under the convention indicated by the indicator element corresponding to the instruction's page.

B1 Cont

2. (amended) The computer of claim 1:

wherein the two architectures are two instruction set architectures;

and wherein the adapting includes controlling instruction execution hardware of the computer to interpret the instructions according to the two instruction set architectures according to the indicator elements.

3. (amended) The computer of claim 1, wherein the two conventions are first and second calling conventions, and further comprising:

hardware and/or software designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first calling convention to a region whose indicator element indicates the second calling convention, and in response to the recognition, to alter the data storage content of the computer to create a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention.

4. (amended) A method, comprising the steps of:

executing instructions fetched from first and second regions of a memory of a computer, the instructions of the first and second regions being coded for execution by computers of first and second architectures or following first and second data storage conventions, respectively, the memory regions having associated first and second indicator elements, the indicator elements each having a value indicating the architecture or data storage convention under which instructions from the associated region are to be executed, the indicator elements being architecturally addressable when the processor pipeline is executing under one of the architectures or data storage conventions, and architecturally unaddressable when the computer is executing under the other architecture or data storage convention;

when execution of the instruction data flows or transfers from the first region to the second, adapting the computer for execution in the second architecture or convention.

---

14. (amended) The method of claim 10, wherein a mode of execution of the instructions is changed without software intervention when execution flows or transfers from the first region to the second.

15. (amended) The method of claim 10, wherein execution of the computer takes an exception when execution flows or transfers from the first region to the second.

---

18. (amended) The method of claim 10, wherein the two conventions are first and second data storage conventions, and further comprising the step of:

recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

---

20. (amended) The method of claim 18, further comprising the steps of:

classifying control-flow instructions of a computer instruction set into a plurality of classes; and

during execution of a program on a computer, as part of the execution of instructions of the instruction set, updating a record of the class of the classified control-flow instruction most recently executed;

the adjusting process being determined, at least in part, by the instruction class record.

B4

21. (amended) The method of claim 18, wherein:

the instruction data coded for execution by a first of the two instruction set architectures observes a data storage convention associated with the first architecture, and instruction data coded for execution by a second of the two instruction set architectures observes a second, different, data storage convention associated with the second architecture, a single indicator element indicating both the instruction set architecture and the data storage convention;

and further comprising the step of recognizing when program execution flows or transfers from a region using the first instruction set architecture to a region using the second instruction set architecture, and in response to the recognition, adjusting the data storage content of the computer from the first storage convention to the second.

22. (amended) A method, comprising the steps of:

executing instructions fetched from first and second regions of a memory of a computer, the instructions of the first and second regions being coded for execution by computers following first and second data storage conventions, the memory regions having associated first and second indicator elements, the indicator elements each having a value indicating the data storage convention under which instructions from the associated region are to be executed;

recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indictor element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

23. (amended) The method of claim 22, further comprising the step of:

overlaying the logical resources of the first and second instruction set architectures onto the physical resources of the computer according to a mapping that assigns corresponding resources of the two architectures to a common physical resource of a computer when the resources serve analogous functions in the calling conventions of the two architectures.

24. (amended) The method of claim 22, wherein the adjusting step further comprises:

altering a bit representation of a datum from a first representation in the first convention to a second representation in the second convention, the alteration of representation being chosen to preserve the meaning of the datum across the change in data storage convention.

Sub C3

27. (amended) The method of claim 22, wherein

a rule for copying data from the first location to the second is determined by examining a descriptor associated with the location of execution before the recognized execution flow or transfer.

33. (amended) The method of claim 28, further comprising the step of:

taking a processor exception in response to the recognition, a handler for the exception programmed to copy a datum from a first location to a second location, the first location having a use under the first data storage convention analogous to the use of the second location under the second data storage convention.

34. (amended) The method of claim 22, further comprising the step of:

classifying control-flow instructions of a computer instruction set into a plurality of classes; and during execution of a program on a computer, as part of the execution of instructions of the instruction set, updating a record of the class of the classified control-flow instruction most recently executed;

[illegible]

the adjusting process being determined, at least in part, by the instruction class record.

---

37. (amended) A computer processor, comprising:

a processor pipeline configured to alternately execute instructions of computers of two different architectures or processing conventions; and

a memory unit designed to fetch instructions from a computer memory for execution by the pipeline, and to fetch stored indicator elements associated with respective memory regions of a single address space from which the instructions are to be fetched, each indicator element designed to store an indication of the architecture or execution convention under which the instruction data of the associated region are to be executed by the processor pipeline, the indicator elements being maintained in storage that is architecturally addressable when the processor pipeline is executing in one of the computer architectures or processing conventions, and architecturally unaddressable when the processor pipeline is executing in the other architecture or convention;

[illegible]

the memory unit and/or processor pipeline further designed to recognize an execution flow or transfer from a region whose indicator element indicates one architecture or execution convention to another.

38. (amended) The computer processor of claim 37, wherein the indicator elements are stored in a table of indicator elements distinct from a primary address translation table used by the virtual memory manager, the indicator elements of the table associated with corresponding pages of the memory.

---

42. (amended) The computer processor of claim 40:

[illegible]

each indicator element being further designed to store an indication of a calling convention under which the instruction data of the associated region are coded for execution by the processor pipeline;

B8 Cont.

and further comprising software programmed to alter the data storage content of a computer using the computer processor to create a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention;

the memory unit further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first calling convention to a region whose indicator element indicates the second calling convention, and in response to the recognition, to invoke the transition management software.

B9

44. (amended) The computer processor of claim 42, wherein the memory unit and software are designed to effect a transition between instruction boundaries, between execution in a region coded in the first instruction set using the first calling convention to execution in a region coded in the second instruction set using the second calling convention, so that code at the source of the flow or transfer may effect the execution transition without being specially coded for code at the destination.

B10

47. (amended) The computer processor of claim 42, wherein

a rule for altering the data storage content from the first calling convention to the second is determined by examining a descriptor associated with the location of execution before the recognized execution flow or transfer.

B11

50. (amended) The computer processor of claim 40, further comprising circuitry designed to raise an exception when the computer recognizes that execution has flowed or transferred from a region whose indicator-element indicates one architecture or execution convention to another.

B12

53. (amended) The method of claim 51, wherein the two architectures are two instruction set architectures, and further comprising the step of:

54. (amended) The method of claim 51, wherein the two conventions are first and second data storage conventions, and further comprising the step of:

recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

[illegible]

55. (amended) The method of claim 54, further comprising the steps of:

storing instruction data in a third page, the instruction data of the third page being coded for execution by one of the two architectures, and observing a data storage convention associated with the other of the two architectures;

storing indicator elements indicating the data storage convention observed by the instructions of the respective pages; and

recognizing each transition of program execution from a page using the first data storage convention to a page using the second data storage convention, and in response to the recognition, adjusting the data storage content of the computer from the first storage convention to the second, and vice-versa.

56. (amended) The method of claim 53, wherein:

the instruction data coded for execution by a first of the two instruction set architectures observes a data storage convention associated with the first architecture, and instruction data coded for execution by a second of the two instruction set architectures observes a second, different, data storage convention associated with the second architecture, a single indicator element indicating both the instruction set architecture and the data storage convention of the associated page;

and further comprising the step of recognizing when program execution flows or transfers from a page using the first instruction set architecture to a page using the second instruction set architecture, and in response to the recognition, adjusting the data storage content of the computer from the first storage convention to the second.

57. (amended) The method of claim 54, wherein the two conventions are a register-based calling convention and a memory stack-based calling convention, and further comprising the step of:

recognizing when program execution has flowed or transferred from a page using the register-based calling convention to a page using the memory stack-based convention, and in response to the recognition, adjusting the data storage content of the computer from the first calling convention to the second.

Sub c7

59. (amended) The method of claim 54, wherein

a rule for copying data from the first location to the second is determined by examining a descriptor associated with the location of execution before the recognized execution flow or transfer.

61. (amended) A microprocessor chip, comprising:

an instruction unit, configured to fetch instructions from a memory managed by the virtual memory manager, and configured to execute instructions coded for first and second different computer architectures or coded to implement first and second different data storage conventions;

the microprocessor chip being designed (a) to retrieve indicator elements stored in association with respective pages of the memory, each indicator element indicating the architecture or convention in which the instructions of the page are to be executed, and (b) to recognize when instruction execution has flowed or transferred from a page of the first architecture or convention to a page of the second, as indicted by the respective associated indicator elements, and (c) to alter a processing mode of the instruction unit or a storage content of the memory to effect execution of instructions in accord with the indicator element associated with the page of the second architecture or convention.

62. (amended) The microprocessor chip of claim 61, wherein the indicator elements are stored in virtual address translation table entries.

63. (amended) The microprocessor chip of claim 61, wherein the indicator elements are stored in a table distinct from a primary address translation table used by a virtual memory manager, the indicator elements of the table being stored in association with respective pages of the memory.

64. (amended) The microprocessor chip of claim 61, wherein the indicator elements are stored in association with respective physical page frames.

65. (amended) The microprocessor chip of claim 61, wherein the indicator elements are stored in association with respective virtual pages.

66. (amended) The microprocessor chip of claim 61, wherein the indicator elements are stored in entries of a translation look-aside buffer.

67. (amended) The microprocessor chip of claim 61, wherein the indicator elements are stored in an instruction cache.

68. (amended) The microprocessor chip of claim 61, wherein a mode of execution of the instructions is changed without software intervention when execution flows or transfers from the first region to the second.

69. (amended) The microprocessor chip of claim 61, the microprocessor chip being designed to raise an exception when execution flows or transfers from the first region to the second;

and further comprising exception handler software programmed to handle the exception by explicitly controlling a mode of execution of the instructions.

---

73 (amended) The microprocessor chip of claim 71:

each indicator element being further designed to store an indication of a data storage convention under which the instruction data of the associated page are coded for execution by the instruction unit;

and further comprising software programmed to alter the data storage content of a computer using the microprocessor chip, to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention;

B16

the microprocessor chip further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, to invoke the transition management software.

76. (amended) The microprocessor chip of claim 71, further comprising hardware and/or software designed:

(a) to retrieve calling convention indicator elements stored in association with respective pages of the memory, each calling convention indicator element indicating which of a register-based calling convention or a memory stack-based calling convention is observed by instructions of the page;

(b) to recognize when instruction execution has flowed or transferred from a page of a memory-based convention to a page of the register-based calling convention, as indicated by the calling convention indicator elements associated with the respective pages, and

B17

(c) in response to the recognition, to alter a storage content of the computer to create a program context under the register-based convention logically equivalent to a pre-alteration program context under the memory-based convention.

77. (amended) The microprocessor chip of claim 61:

wherein the two conventions are first and second data storage conventions;

and further comprising software programmed to alter the data storage content of a computer using the microprocessor chip, to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention;

the microprocessor chip being further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indicator element indicates the second data storage convention, and in response to the recognition, to invoke the transition management software.

---

84. (amended) The microprocessor chip of claim 79, further comprising:

software and/or hardware designed to copy a datum from a third location to a fourth, the third location having a use under the first data storage convention analogous to the use of the third location under the first data storage convention and to the fourth location under the second data storage convention, the software and/or hardware for the copying being programmed to assume that exactly one of the first and third locations is no longer required by the execution of the program.

85. (amended) The microprocessor chip of claim 61, further comprising hardware and/or software designed:

(a) to retrieve calling convention indicator elements stored in association with respective pages of the memory, each calling convention indicator element indicating which of a register-based calling convention or a memory stack-based calling convention is observed by instructions of the page;

(b) to recognize when instruction execution has flowed or transferred from a page using the register-based calling convention to a page using the memory stack-based convention, as indicated by the calling convention indicator elements associated with the respective pages, and

(c) in response to the recognition, to alter a storage content of the computer to create a program context under the memory-based convention logically equivalent to a pre-alteration program context under the register-based convention.

---

87. (amended) A method, comprising the steps of:

executing a control-transfer instruction under a first execution context of a computer, the instruction being architecturally defined to transfer control directly to a destination instruction for execution in a second execution context of the computer;

before executing the destination instruction, altering the data storage content of the computer to establish a program context under the second execution context that is logically equivalent to the context of the computer as interpreted under the first execution context, the reconfiguring including at least one data movement operation not included in the architectural definition of the control-transfer instruction.

---

92. (amended) The method of claim 87, further comprising the step of:

altering a bit representation of a datum from a first representation in the first context to a second representation in the second context, the alteration of representation being chosen to preserve the meaning of the datum across the change in execution context.

Sub C10

94. (amended) A method, comprising the steps of:

executing a section of computer object code twice, without modification of the code section between the two executions, the code section materializing a destination address into a register and being architecturally defined to directly transfer control indirectly through the register to the destination address, the two executions materializing two different destination addresses;

the two destination code sections at the two materialized destination addresses being coded in two distinct instruction sets, neither instruction set being a subset of the other.

---

96. (new) A microprocessor chip, comprising:

two instruction decoders designed to decode instructions of first and second instruction sets, respectively, and circuitry of a single instruction pipeline designed to execute the instructions decoded by either of the two instruction decoders;

circuitry and/or software designed to detect when execution flows or transfers control from code coded in one instruction set to code coded in the other, program code in the first and second instruction sets using first and second different data storage conventions, respectively; and

circuitry and/or software designed to respond to the detection by altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

Sub C11

97. (new) The computer processor of claim 96, wherein the memory unit and software are designed to effect a transition between instruction boundaries, between execution in a region coded in the first instruction set using the first calling convention to execution in a region coded in the second instruction set using the second calling convention, so that code at the source of the flow or transfer may effect the execution transition without being specially coded for code at the destination.

98. (new) The microprocessor chip of claim 96, wherein the two conventions are two calling conventions.

99. (new) The computer processor of claim 98, wherein:

one of the two calling conventions is a register-based calling convention, and the other calling convention is a memory stack-based calling convention.

B2 cont

Sub C12

100. (new) The microprocessor chip of claim 96, further comprising:

software and/or hardware designed to copy a datum from a first location to a second location, the first location having a use under the first calling convention analogous to the use of the second location under the second calling convention.

101. (new) The microprocessor chip of claim 100, further comprising:

software and/or hardware designed to copy a datum from a third location to a fourth, the third location having a use under the first data storage convention analogous to the use of the third location under the first data storage convention and to the fourth location under the second data storage convention, the software and/or hardware for the copying being programmed to assume that exactly one of the first and third locations is no longer required by the execution of the program.

Sub C13

102. (new) The computer processor of claim 96, wherein

a rule for altering the data storage content from the first calling convention to the second is determined based on an instruction at the location of execution at the source of the recognized execution flow or transfer.

Cont C13

103. (new) The computer processor of claim 96, wherein
a rule for altering the data storage content from the first calling convention to the second is determined by examining a descriptor associated with the location of execution before the recognized execution flow or transfer.

[illegible] Cont

104. (new) A method, comprising the steps of:
executing instructions fetched from first and second regions of a single address space of the memory of a computer, the instructions of the first and second regions being coded for execution by computers of first and second architectures or following first and second data storage conventions, respectively, the memory regions having associated first and second modifiable indicator elements, a hardware structure for storing the indicator elements enforcing a requirement that the memory regions be necessarily disjoint, the modifiable indicator elements each having a value indicating the architecture or data storage convention under which instructions from the associated region are to be executed;
when execution of the instruction data flows or transfers from the first region to the second, adapting the computer for execution in the second architecture or convention.

105. (new) The method of claim 104, wherein:
the regions are pages managed by a virtual memory manager.

106. (new) The method of claim 105, wherein the modifiable indicator elements are stored in a table, each modifiable indicator element associated with a corresponding physical page frame.

107. (new) The method of claim 105, wherein the entries are entries of a translation look-aside buffer.

108. (new) The method of claim 104:
wherein the two architectures are two instruction set architectures;

and wherein the adapting step includes controlling instruction execution hardware of the computer to interpret the instructions according to the two instruction set architectures according to the modifiable indicator elements.

109. (new) The method of claim 108, wherein:

one of the regions stores an off-the-shelf operating system binary coded in an instruction set non-native to the computer, the non-native instruction set providing access to a reduced subset of the resources of the computer.

B22 cont.

110. (new) The method of claim 108, wherein the two conventions are first and second data storage conventions, and further comprising the step of:

recognizing when program execution has flowed or transferred from a region whose modifiable indicator element indicates the first data storage convention to a region whose modifiable indicator element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

111. (new) The method of claim 104, wherein the two conventions are first and second data storage conventions, and further comprising the step of:

recognizing when program execution has flowed or transferred from a region whose indicator element indicates the first data storage convention to a region whose indictor element indicates the second data storage convention, and in response to the recognition, altering the data storage content of the computer to create a program context under the second data storage convention that is logically equivalent to a pre-alteration program context under the first data storage convention.

112. (new) The method of claim 111, wherein:

one of the two data storage conventions is a register-based calling convention, and the other data storage convention is a memory stack-based calling convention.

113. (new) A computer processor, comprising:

a processor pipeline configured to alternately execute instructions of computers of two different architectures or processing conventions; and

a memory unit designed to fetch instructions from a computer memory for execution by the pipeline, and to fetch stored indicator elements associated with respective necessarily-disjoint memory regions of a single address space from which the instructions are to be fetched, each indicator element designed to store an indication of the architecture or execution convention under which the instruction data of the associated region are to be executed by the processor pipeline;

the memory unit and/or processor pipeline further designed to recognize an execution flow or transfer from a region whose indicator element indicates one architecture or execution convention to another.

B3 Cont

114. (new) The computer processor of claim 113, wherein the two architectures are two instruction set architectures, and further comprising:

processor pipeline control circuitry designed to control the processor pipeline to effect interpretation of the instructions under the two instruction set architectures alternately, according to the associated indicator elements.

115. (new) The computer processor of claim 114, further comprising:

software programmed to manage a transition between the execution of a program executing in the first instruction set architecture, being an instruction set architecture native to the computer processor, and execution of an off-the-shelf operating system coded in the second instruction set, being an instruction set non-native to the computer, providing access to a reduced subset of the resources of the computer.

116. (new) The computer processor of claim 114:

each indicator element being further designed to store an indication of a calling convention under which the instruction data of the associated region are coded for execution by the processor pipeline;

and further comprising software programmed to alter the data storage content of a computer using the computer processor to create a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention;

the memory unit further designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first calling convention to a region whose indicator element indicates the second calling convention, and in response to the recognition, to invoke the transition management software.

117. (new) The computer processor of claim 116, wherein the memory unit is designed to recognize a single indicator element to indicate both the instruction set architecture and calling convention of a region.

[illegible]

118. (new) The computer processor of claim 113, wherein the two conventions are first and second calling conventions, and further comprising:

hardware and/or software designed to recognize when program execution has flowed or transferred from a region whose indicator element indicates the first calling convention to a region whose indicator element indicates the second calling convention, and in response to the recognition, to alter the data storage content of the computer to create a program context under the second calling convention that is logically equivalent to a pre-alteration program context under the first calling convention.

119. (new) The computer processor of claim 118, wherein the memory unit and software are designed to effect a transition between instruction boundaries, between execution in a region coded in the first instruction set using the first calling convention to execution in a region coded in the second instruction set using the second calling convention, so that code at the source of the flow or transfer may effect the execution transition without being specially coded for code at the destination.

120. (new) The computer processor of claim 118, wherein the two conventions are two calling conventions.

121. (new) The computer processor of claim 118, wherein:

one of the two calling conventions is a register-based calling convention, and the other calling convention is a memory stack-based calling convention.

122. (new) The computer processor of claim 118, wherein the logical resources for support of the first and second instruction set architectures are overlaid on the physical resources of the computer processor according to a mapping that assigns corresponding resources of the two architectures to a common physical resource when the resources serve analogous functions in the calling conventions of the two architectures.

[illegible]

123. (new) The computer processor of claim 118, further comprising:

a transition manager designed to effect a transition between execution under the first calling convention to execution under the second calling convention, the transition manager designed to alter a bit representation of a datum from a first representation to a second representation, the alteration of representation being chosen to preserve the meaning of the datum across the change in calling convention.

124. (new) The computer processor of claim 118, further comprising:

software and/or hardware designed to copy a datum from a first location to a second location, the first location having a use under the first calling convention analogous to the use of the second location under the second calling convention.

125. (new) The computer processor of claim 124, further comprising:

software and/or hardware designed to copy a datum from a third location to a fourth, the third location having a use under the first calling convention analogous to the use of the third location under the first calling convention and to the fourth location under the second calling convention, the software and/or hardware for the copying being programmed to assume that exactly one of the first and third locations is no longer required by the execution of the program.

Sub C14

126. (new) The computer processor of claim 118, wherein

a rule for altering the data storage content from the first calling convention to the second is determined by examining a descriptor associated with the location of execution before the recognized execution flow or transfer.

127. (new) The computer processor of claim 118, wherein control-flow instructions of the instruction set are classified into a plurality of classes; and

the processor pipeline updates a record of the class of the classified control-flow instruction most recently executed;

the storage alteration process being determined, at least in part, by the instruction class record.

[illegible]

128. (new) The computer processor of claim 113, wherein:

the regions are pages managed by a virtual memory manager.

129. (new) The computer processor of claim 113, wherein the indicator elements are stored in virtual address translation table entries.

130. (new) The computer processor of claim 113, wherein the indicator elements are stored in a table of indicator elements distinct from a primary address translation table used by the virtual memory manager, the indicator elements of the table associated with corresponding pages of the memory.

Sub C15

131. (new) The computer processor of claim 113, wherein the indicator elements are stored in respective entries of a table whose entries are associated with corresponding physical page frames.

132. (new) The computer processor of claim 113, wherein:

the indicator elements are stored in storage that is architecturally addressable when the processor pipeline is executing in one of the computer architectures or processing